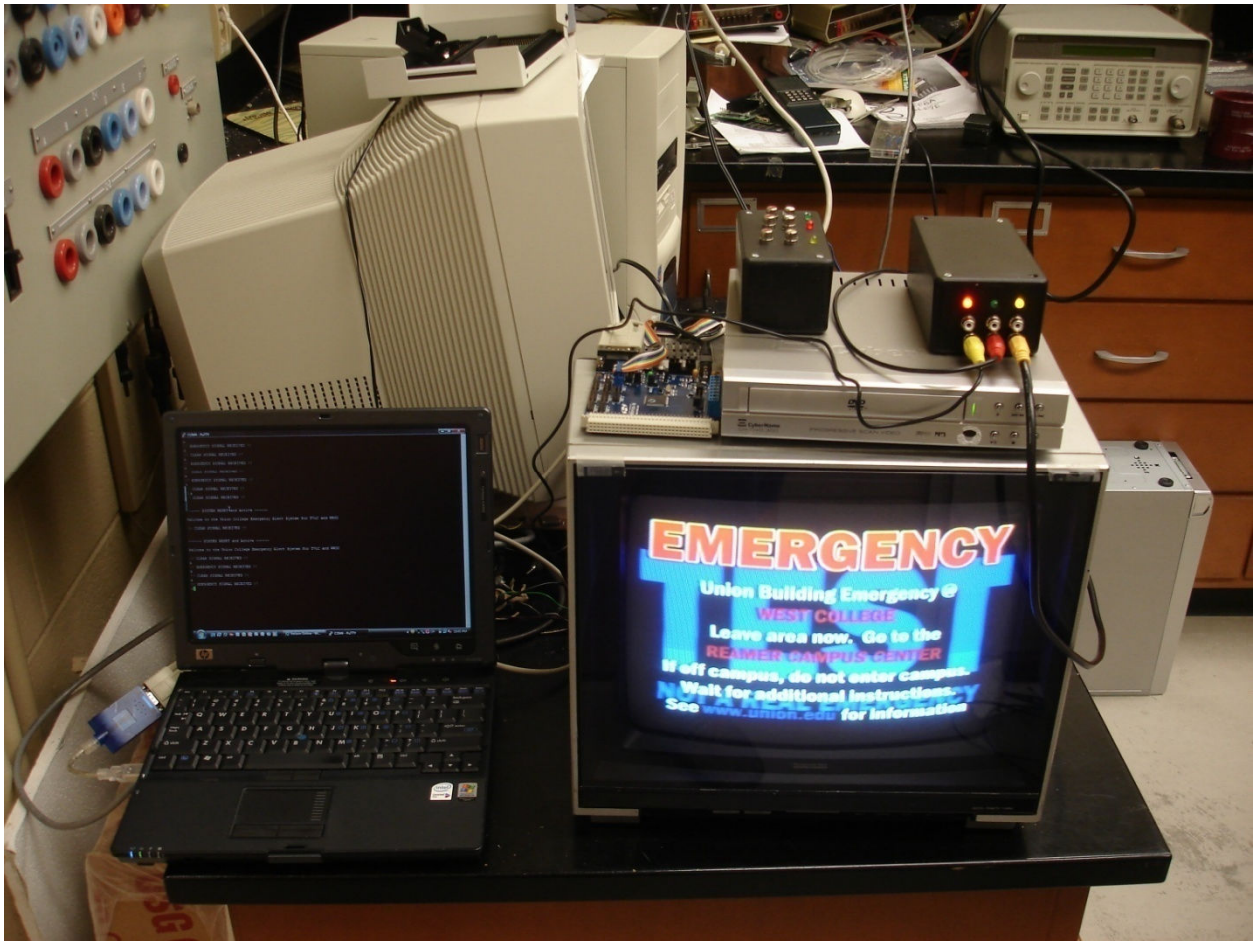
Emergency Alert System

For Union College Television and Radio



Dave Swift

Senior Capstone Project

ECE-499

Winter Term 2008

April 9, 2008

Advisor: Prof. James Hedrick

Report Summary

This report is designed to give the reader an overall, yet detailed description my senior capstone design project. At Union College, all senior Computer Engineering students must complete "a project of their own choosing. Students work in small teams or on their own and alongside a faculty member on their project. The project spans a total of 30 weeks…" (ECE Web site) where students design a project and implement it. For my particular project, I have chosen to implement an Emergency Alert System (EAS) for Union College's television and radio stations, TVuC and WRUC, respectively. This comes on the request from Union's Campus Safety department, who had plans regardless of this project to implement such a system. This project includes both software and hardware prototypes, as both are necessary for the system to function completely.

This project allows for members of Union College's Campus Safety department to activate the system from computers with the client software pre-installed, select a pre-recorded emergency scenario from a list, and remotely "take over" both the TV and radio stations by displaying this emergency information to the campus. This system will help the college get pertinent information out to the campus as part of their ongoing reevaluation of the campus-wide emergency plan, also consisting of SMS-based text messaging to those who have opted in, campus-wide emails, and updates to the college's home Web page. As the president and technical director of TVuC, the campus's television network, I was approached with the idea to implement such a design, and decided to do this for this particular design project, which further shows the real-world relationship between a practicing engineer and their client.

Much of the software portion of this project pertaining to the client, server, and scripting, was designed and written by my former project partner, Zach Smith, who had become ill halfway through the Winter term, and has dropped out from the remainder of the project. Therefore, the final

implementation of the software and details about the aspects of said software are minimal, as he was unable to complete the implementation for the project due date. Therefore, for the remainder of this paper, we will assume that the software has been implemented appropriately so that the system fully functions. In the future, I hope that this is actually completed either by him or by a new student, so that our client is indeed satisfied with the implementation.

Table of Contents

Report Summary 02

Table of Contents 04

Introduction 05

Background 07

Design Requirements 10

Design Alternatives 13

Final Design and Implementation 15

Performance Estimates and Results 22

Production Schedule 26

Cost Analysis 27

User's Manual 29

Discussion, Conclusions, and Recommendations 30

References 33

Appendices 34

Introduction

For this capstone project, I have designed an emergency alert system (EAS) for Union's on-campus television station, TVuC, and radio station, WRUC. This allows members of Union College's Campus Safety department to activate the system from computers with the client software pre-installed, select a pre-recorded emergency scenario from a list, and remotely "take over" both the TV and radio stations by displaying this emergency information to the campus. This system will help the college get pertinent information out to the campus as part of their ongoing reevaluation of the campus-wide emergency plan, also consisting of SMS-based text messaging to those who have opted in, campus-wide emails, and updates to the college's home Web page. As the president and technical director of TVuC, the campus's television network, I was approached with the idea to implement such a design, and decided to do this for this particular design project, which further shows the real-world relationship between a practicing engineer and their client.

The overall objective of this project is to allow authorized users to easily and quickly get emergency scenario information on the TV and radio should an emergency arise. In order to accomplish this, switches needed to be built for both the TV and radio components, and software needed to be written to control the system. The switch for TV forces the audio and video outputs from the default state, to the output of a computer containing emergency scenario information. The switch for radio does the same, but instead of taking in audio and video, it simply takes in stereo audio. In addition to the switches, a computer was dedicated to the project, which contains the prerecorded emergency scenarios as well as the server software which will launch the video and activate the hardware switches. The emergency scenario information is stored as MPEG video files; however the video simply contain a single frame containing written text displaying the message along with an audio clip of the scenario. Therefore, when the user selects a specific scenario, the computer will automatically begin playing the video file as well as transmitting an output command to the necessary hardware switches. Thus, when

the system is active, the TV station displays a “slide” of text, along with audio accompaniment, and the radio station simply broadcasts that audio component. This way, anyone watching the TV station or listening to the radio station will see and/or hear the important information during emergency alerts.

This report is broken down in a way which gives the reader an overall feeling for the project, then goes into design specifications and technical specifics. Potential alternative implementations are then presented, along with reasons why they were not chosen instead. Results are then discussed, and any additional information relating to the project, including the production schedule, detailed allotment of costs, and references are given. In addition, a User’s Manual is provided, which will be presented to Campus Safety for their training purposes and will be referenced by them if they run into trouble launching the system. Finally, an appendices section is included, containing the source code for all software written exclusively for this project.

Background

After the April 16, 2007 massacre at Virginia Polytechnic Institute, college campuses across the nation have begun to re-explore their emergency scenario plans. At Union, campus officials have decided to implement a better system of notifying students, faculty and staff should a situation on our campus arise. This includes notification via the college's television and radio stations, as well as SMS-based text messaging to those who have opted in, campus-wide emails, and updates to the college's home Web page. As the president of the television studio, TVuC, I was approached by the director of Campus Safety, Bill Sickenger, in the spring of the 2006-2007 academic year, with hopes that I could implement the television component for them. Adding the radio component would only take little more effort than simply the television portion, so I decided that I would indeed be able to implement some form of emergency alert system (EAS). However, due to the complexity of a successfully working system, I was unable to simply do this in my "spare time". It then dawned on me that I could implement the system for this capstone design requirement, as it would not only accomplish the goals and requirements of such as project, it would actually have an engineer/client component to it. Now that the project is complete, I feel that I have been able to assist the college in accomplishing their goal for alerting the campus when a situation occurs.

There are many aspects of this project that deal directly with society. Most importantly, this project deals with the health and safety of those on campus, as well as the social and ethical impact. Since anytime an emergency situation occurs, communication among officials must be planned out and well implemented, preventing an even worse situation from happening. Therefore, this project will produce a product which is a tool for dealing with the situation. It will not be my responsibility to come up with the scenarios, as what a student thinks might be an emergency may not in fact call for such alarm, and vice versa. In addition, it will not be up to me to "activate" the system for the same reasons. If I did, it may cause for a false-alarm, and get people confused causing chaos. In addition, if the system

is activated, but the user selects the wrong message, the same can occur. The last thing I would want is to alert the campus to a building fire when a mass shooter is present in a different location. Therefore, the system needs to be secure enough that it cannot be hacked into or changed by anyone other than those authorized by college officials.

Finally, there are economic and manufacturability concerns that must be addressed. There are currently commercial products available on the market that can accomplish this same task, albeit individually for both TV and radio. However, their implementation is much more complex and the cost is much higher than what the college wishes to implement. The cost for this project is very minimal. The software used in this project is going to be designed in house, so there is no cost associated with that. In addition, there are already audio links between the radio and TV stations, so there is no cost there. The main expenditure of the project is in the computer and switches which will control the system. The requirements for the computer are quite basic—any system with sufficient hard drive space, which is supported by the Linux platform, and has TV-out capabilities, is sufficient for functioning as the server, and is fairly inexpensive. The switches that override current broadcasting with emergency information are also inexpensive, as the main cost comes from the 8051 microcontroller which will be used to activate the switches when the system is activated. The software coding which will run the video clips and activate the system will also be programmed in house, so no outside software will really be required to run the system. Therefore, cost will be as low as possible while the benefits of this project will be quite high.

Prior to this past fall term when the actual design and implementation of this project came to fruition, I worked with my client to implement an alternative system, which served as a temporary solution and as a backup system, should this new system ever fail. The alternative system is not nearly as user friendly as my system; however, it does allow emergency information to be presented to the

campus community. Campus Safety has purchased, upon the recommendation of the TV station, a Leightronix Mini-T-NX Digital Video Server. This device functions with two major rolls. Not only does it serve as a backup for storing emergency alert information, it is our main video head-end controller, in which all broadcasts for the television station pass through. It contains multiple inputs, all of which are RCA audio and video, and passes the output to both our video modulator (which converts the A/V signals and broadcasts them over our CATV network) and a monitor for sampling before broadcasts go live. In our case, we have multiple DVD players, a VCR, and a computer all hooked up to this controller, and both audio and video from these devices pass through the controller.

However, what makes this controller special is the fact that not only can it control these devices, it also contains an internal hard drive which can store MPEGs and JPEGs, and can be accessed anywhere on campus over a TCP/IP connection, so long as the proprietary software is preinstalled. This means that we can store the emergency scenario slides on this device, in which Campus Safety could use should the new system fail. One may wonder why this other system is necessary. The simple answer to this is its software is quite cumbersome and unintuitive and would be nearly impossible to use by an untrained user. In addition, only a user with administrative rights can update the device, should other scenarios need to be added. Thus, it will function as a backup system going forward, which was deemed necessary by my client.

Design Requirements

To fully understand the requirements for this system, specifics for television and radio have to be recognized, as all audio and/or video for both TV and radio will be passing through the EAS. For the entire system to work, it is assumed and required that electricity and internet are available to the television and radio studios, located in the Reamer Campus Center on the Union College Campus. Obviously without electricity, the entire cable and radio systems would be non-functional, as they are currently set up without backup generators. If a reliable connection to the internet were to become unavailable, a manual activation can be completed from the emergency server, located in the television studio. The client software will run alongside the server software, so the interface is no different to the user if they were to launch it from this machine, which would not require an internet connection. If neither option is viable, the emergency system will fail to activate; a design limitation that is acceptable to my client. In addition, a physical audio link is required between the emergency computer and the television and radio broadcasting equipment. Since both of the studios are adjacent to each other, a simple audio out connection from the television station to the radio station has already been established between the two rooms and will be used with the EAS. This simplifies the deliverance of the audio from the emergency computer to both media outlets.

The entire system will be controlled primarily by a dedicated computer running Ubuntu, a distribution of the open-source Linux operating system. Linux has been chosen since it is open source, making the cost nothing, while reliability is phenomenally high. Requirements for the computer include the fact that it must be running 24/7, and must not automatically restart, perform system updates, or do anything else that would interfere with the operation of this system, which other operating systems may do. In addition, Linux works natively with C++ which is the language the server software has been written in. The computer must also be able to activate the hardware switches via a serial out connection to an 8051 microcontroller. Finally, the computer will serve as a media server, containing

the necessary files for each scenario presented to the user, and output the necessary video and audio when activated through its TV-out and audio-out connections.

To simplify the process and reduce the chance of an error occurring, it makes the most sense to store the text and video in simple uncompressed MPEG video files. For each emergency scenario, a separate file exists. Each file contains a simple slide of fixed text to display on the screen, along with an audio track containing a recording of the message being broadcasted, some with additional information that cannot fit on the screen. The messages have been provided by Campus Safety. The length of the file will be determined based on the length of the audio track, and when it ends, the entire clip will loop automatically, defined by a setting in the MPEG player native to Linux. The MPEG video file can also be played natively by Linux, so no special software will be required for this operation to run.

Based on the requests of my client, the finished product must easily be accessible if and when it needs to be activated. Upon reviewing possible implementations to accomplish this, such as using proprietary software and/or hardware, it made the most sense to have unique software installed on the client side to activate the system. This client software is coded in Java, so cross-platform compatibility is the greatest. Therefore, the system can only be launched on systems that Campus Safety deemed acceptable. The software has a hard-coded IP address and port mapping to the server, and will NOT be accessible outside the campus. This is done to prevent people who have obtained the software from otherwise hacking the system and launching it when it is not appropriate. In addition, the server will require user authentication in the form of a user name and one complex password. Upon logging in, the user will be presented with all scenarios available in a simple text or button interface, as well as a reset interface to resume normal programming once the emergency is over. This is done for simplistic purposes and to allow new messages to be displayed without having to recompile and rebuild the software each time. Each scenario references the corresponding video file and passes this information

via a script into the media player which launches the correct video slide in full screen on the TV out port of the computer and sends audio to both the television station's modulator and radio station output.

The television and radio stations broadcast nearly 100% of the time. Therefore, a system had to be designed that would pass the normal broadcasting through without any loss of signal, yet be able to interrupt it when an emergency message needs to be sent through. Thus, a switch has been designed that will pass the normal broadcasting through unless it is "switched" via the emergency computer. In this case, it will output the emergency messages until reset by an authorized user in which normal broadcasting will resume. There are two switches used in this project—one for television, and one for radio.

Finally, Campus Safety has requested that a backup plan be available, should some component of this system fail. A backup plan has been implemented for the television station, which is based on the studio's Leightronix Mini-T-NX Digital Video Server, which functions as the main controller for all of the station's video sources. Since this device can be accessed via a TCP/IP connection, and contains an internal hard drive which can store MPEGs and JPEGs, emergency scenario slides can be stored on this device, in which Campus Safety could use should the new system fail. Since the software quite cumbersome and unintuitive and would be nearly impossible to use by an untrained user, it will function solely as a backup system.

Design Alternatives

When looking at potential design alternatives, slight modifications could have been implemented to the actual design, or completely different implementations could have been used. For example, the client software could have been coded in C++ instead of Java, but platform compatibility would have been compromised. In addition, the client/server software could have been Web based, yet the security would have been much less, even with encryption. As far as launching the emergency messages, an alternative possibility was to use still slides with separate audio clips. However, this would have been much more complicated and would have required more programs to be launched when the system is activated. Microsoft PowerPoint was considered for doing this, with audio implemented in the background of each slide. Finally, a different type of scripting interface could have been used to launch the system. The alternatives and combinations of getting the message out are endless, and I feel confident that what has been chosen will be the most efficient method of implementation for this system.

The hardware could have been implemented many different ways, some of which depend on the software which is used. A pre-manufactured switch could have used instead of our custom switches. However, finding one that interacted with the computer, which is a requirement, was not only nearly impossible, but extremely expensive, and required additional proprietary software to control it. Using the switches that were custom built made this much easier, more secure, and far less expensive.

Another alternative would have been using a premade hardware controller. In the television studio, one already exists to store and launch programming. This device can launch various slides and could have been used to implement this project. However, it would have required a great deal of client training with its proprietary software, and still would not have been as clean as the actual system implemented. In fact, getting it to communicate with the radio station would have required additional

hardware, which would have added additional costs. The controller requires a user to log into its proprietary software, pause current programming, locate the emergency video file, and output it. Resetting the controller once the situation is over is just as hard. Such a device is quite expensive as well, and to dedicate them entirely to an emergency alert system would be cost inefficient. That being said, I have experimented slightly with this, and have in fact used this as a temporary solution for Campus Safety to display a simple slide on the television, alerting viewers to log onto the college's Web site for more information until the final implementation was completed. Only a single member of Campus Safety got trained, and still had trouble learning the system. Luckily, it never needed to be activated in the short time it was in use.

Another alternative would be to purchase two separate emergency alert systems designed specifically for television or radio. This would accomplish the goals of the project; however, the cost in doing this would be ridiculously high. Many implementations for television require cable-company approval or implementation, which would be out of our reach. Finally, certain mandates that may not necessarily apply to a campus like ours may be implemented by governing bodies, such as the FCC, again boosting the complexity and overall cost. Those provided by the cable-company may require a monthly fee, which can easily be avoided with my system.

Final Design and Implementation

The final implementation of this project includes custom designed software and hardware which make an emergency alert system. When correctly implemented, the system allows a user from Union College's Campus Safety department who is authorized by them to launch client software pre-installed on their computer systems to log into the system, select a pre-recorded emergency scenario, also determined by them, which then sends a command to the emergency computer which launches the appropriate message's MPEG file which outputs the video of the message over the computer's TV-out controller and audio through the computer's sound card. In addition, the computer sends an activation command to the system's 8051 microcontroller, which flips the inputs in both the TV station and radio station's respective emergency switches, which then forces the output on both ends to the audio and/or video coming from the emergency computer. The process is repeated when the scenario is over, with the user selecting the "clear system" message which ends the MPEG display and resets both hardware switches to the default state, resuming original programming.

The software that will operate this project is broken down into four aspects, including the client software, the server software, the scripts to launch the MPEG files, and the software which runs the logic of the 8051 microcontroller. As mentioned earlier, the implementation of the client software was done in Java, to allow the most compatibility across different platforms that may be used. With Java, if the client is using Windows, Mac, or Linux hardware, the same build of the software will work across all platforms. Upon launching the software, the client is presented with a prompt asking for their user name and password, and once authenticated, the client software will request from the server a list of all available MPEG files. This was done to allow the client to see newly added scenarios without having to recompile any or all pieces of software. The client then selects the scenario they wish to display, and a command is sent to the server to perform the remaining operations. The only other function of the

client software is to send the “all-clear” message to restore the system to the default state. This functions just like any other scenario on the client side.

The server software is fairly simple as well. The server waits in an always ready state, looking for the activation command from the client. Once a command is received, the server then launches the appropriate script to actually launch the MPEG file on the secondary display of the computer, which is connected to the TV-out, and in turn, displays the message on the TV. In addition, the server software sends an “emergency scenario activation” command over the computer’s serial port which is connected to the 8051 microcontroller. Once both the MPEG is launched, and the 8051 enable command is set, the server software returns to the ready state, waiting for another command, be it another emergency message, or a “clear-all”. If the special “clear-all” message is selected, the server software launches a script to close out the MPEG file, and resets the 8051 to its default ready state. The scripting functions are built into the server software. The same script is used for all scenarios, except the “clear-all”, and the file name is plugged dynamically into the script, so that the appropriate message is displayed. The “clear-all” script simply closes the media player which ends the broadcasting of the scenario message.

The last aspect with regards to software is the logic that controls the 8051 microcontroller. The software implemented here is also quite straightforward. The 8051 is programmed using the C language, using configuration modules that were pre-created by the development board’s Configuration Wizard. Configuration was necessary to allow the 8051 to receive commands from its RS-232 port, as well as the configuration of PORT 0, which is used to interface with the hardware. Two pins, pin 0 and 2 on PORT 0 were configured to switch between high and low to control the state of the hardware switch, with pins 1 and 3 set low, to function as ground ports. Pins 0 and 1 control the switch for the TV station, while pins 2 and 3 control the switch for the radio station.

The logic written to the 8051 does the following: upon powering up or resetting the 8051, perform the necessary configuration commands so that the 8051 does what is mentioned above, then enter the default "ready" state. In this state, the 8051 waits for the character "a" (for activate) to be sent over the serial connection. If the 8051 receives the character "a", it then launches the "emergency scenario" state. In this state, pins 0 and 2 are switched high so that a voltage is outputted from the microcontroller to the switches. In addition, the diagnostic board's LED is lit to show that the 8051 is indeed in the emergency state. While in this state, it waits for any other character to come over the serial connection, in which it will return to the default state, which in turn resets the hardware switch. Using a simple character to enable the switches was chosen such that the server can send multiple activation commands without flip flopping between ready and emergency, just to make sure that the command went through successfully. So, if the 8051 receives five activation commands, it is the same as a single command. The same applies to the clear command. The code used to implement this logic in the 8051 can be seen in the appendices.

The hardware implementation involves two aspects, the emergency computer and the emergency switches (see Figure 01, below). The design implemented calls for a computer to function as the emergency server, containing the server software, the MPEG launch scripts, and the MPEG files. For the system to be as stable as possible, and have the most uptime, Ubuntu, a build of the Linux operating system was chosen which provides a much higher uptime compared to Windows, and to an extent, Mac OSX. With Windows, the system may lockup, do an automatic update, or display a reminder pop-up, all of which could interfere with the display of the MPEG slides. Since the computer must run virtually unattended, Linux will do exactly what is desired. The special requirements for the computer include a hard drive large enough to store each MPEG file (each of which is only a few megabytes), and has TV and audio out capabilities. These requirements are not that great, so I chose to build a new system with a Celeron-D based processor, a gigabyte of memory, a PCI-express based graphics adapter with TV-out,

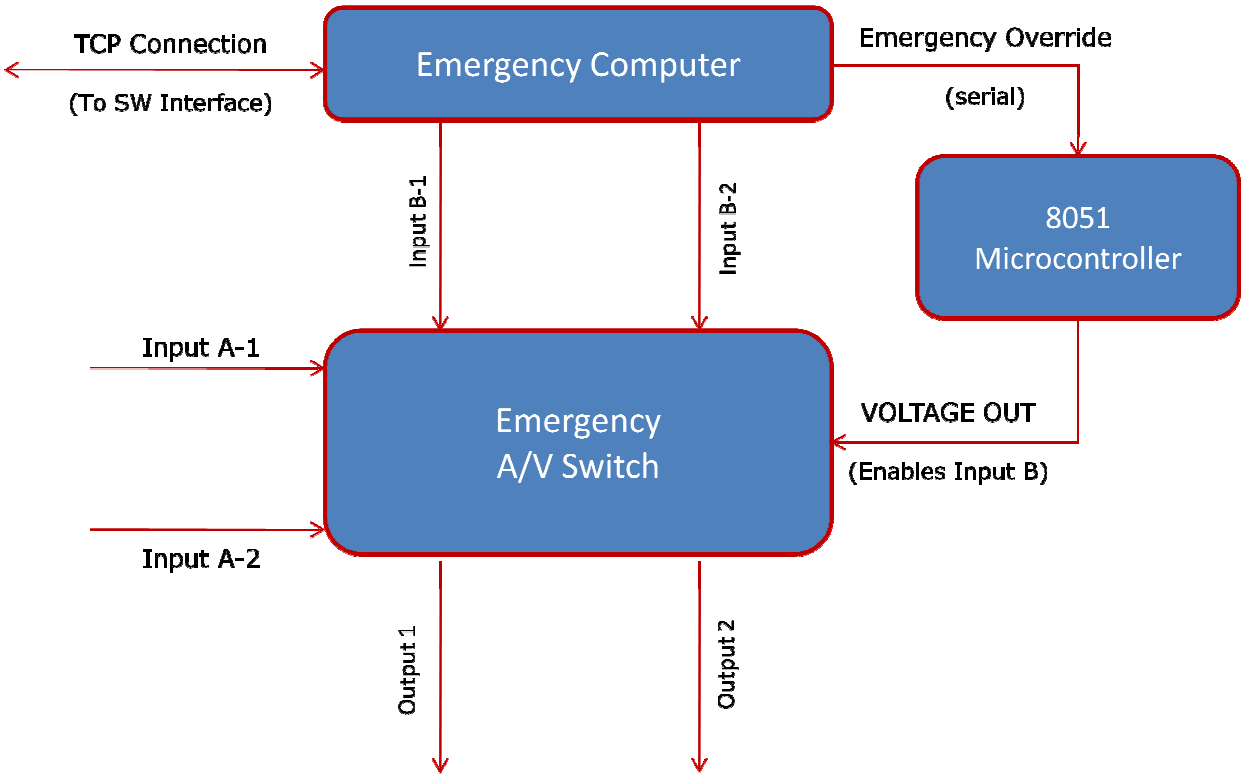
and a 160gb hard drive, all of which are well above the required specifications, but still cost efficient, to allow the system to run smoothly without ever over-taxing the machine.

Figure 01: Hardware Switch

The real "home-made" hardware lies in the form of the switches which change the broadcasting from normal operations to emergency operations. It made the most sense to build two switches, each of which are identical, one being housed in the television studio to control the television broadcast, and one housed in the radio studio to control the radio broadcast. Each switch takes in two inputs, input A and B, with each input containing two channels, channel 1 and channel 2, and outputting either input A or B. Figure 02, below shows the basic setup for the IC. In the case of the TV switch, channel 1 is audio, and channel 2 is video, while with the radio, channel 1 is left audio, and channel 2 is right audio. In all cases, the physical input jacks are RCA. This is sufficient, since the TV station functions using mono audio only, while the radio station does not need any video to be passed. Input A, the default input,

connects to the normal broadcasting, while Input B connects to the emergency broadcasting. Each switch contains a power LED showing when the switch is on (no signals will be passed without power) as well as a LED for each input to show which input is currently active.

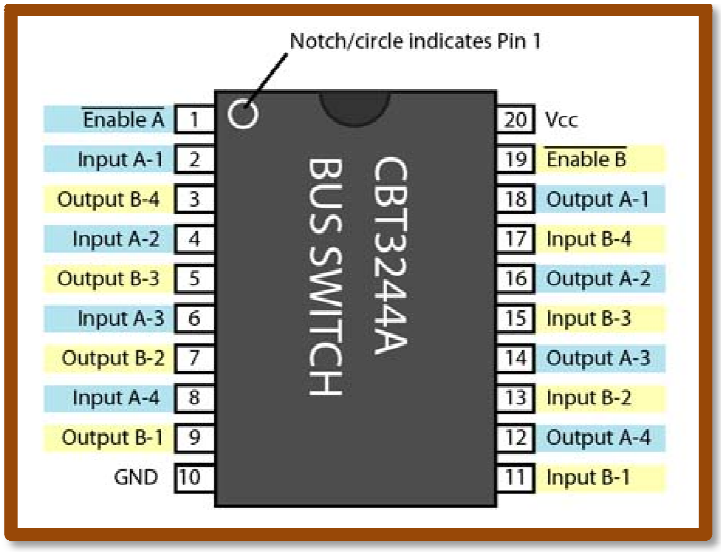
Figure 02: Emergency System Overview



Each switch is connected to a 5V power supply. This power supply runs the LEDs, as well as an IC, which controls the inputs and outputs, as well as a relay to switch the inputs from one to another. The IC is a CBT3244A, which can function for many purposes. In the case of the switch, the IC is used to send either Input A or Input B out to the output switch, based on which enable is selected on the IC. Both enables are active low, so when the voltage drops, the input is enabled. Output A and B are crossed, and send signal to the same set of output jacks, since in this specific usage, only one input will

ever be enabled at a given time, even though for other purposes they both could be. Figure 03, below, shows the set up of the IC bus switch.

Figure 03: Diagram of CBT3244A IC



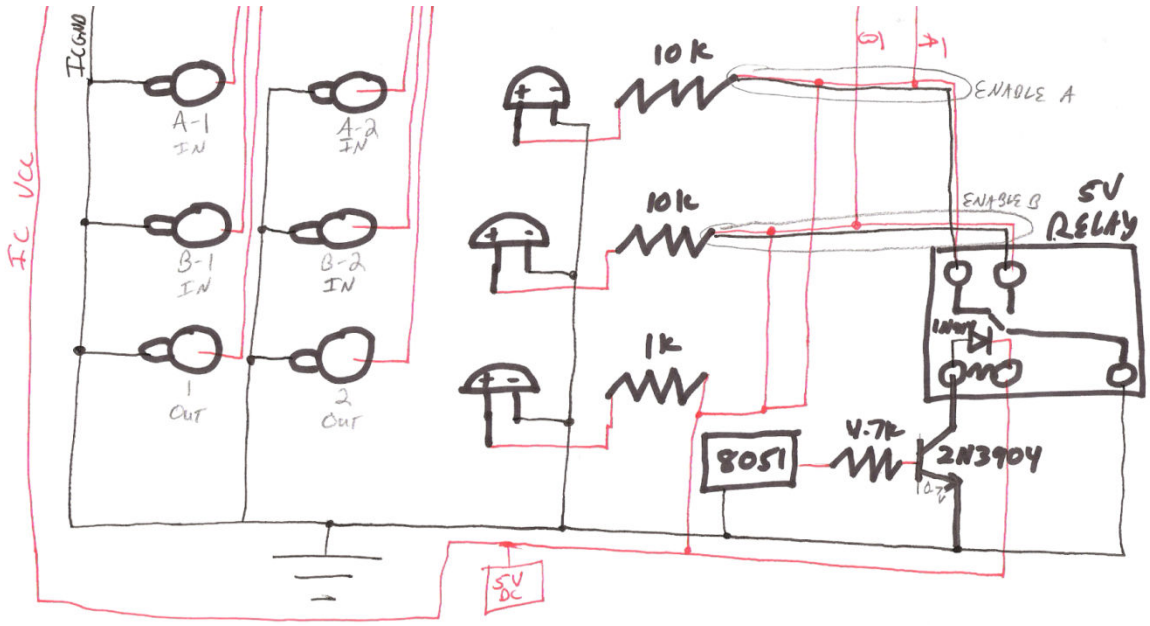
Each switch contains a single pull, double throw (SPDT) relay. The relay connects each input to the output. When the relay is in its default, unpowered state, Input A is passed to the output ports, but when a voltage of 5V is applied to the relay, it switches to passing Input B, effectively disabling Input A. The relays are connected to the 5V power source inside the switch, with the ground connection running to the 8051 microcontroller. When the microcontroller is put into the emergency state, the relay's coil is closed circuited, allowing the relay to switch from Input A to Input B. Finally, a diode was installed across the relay's coil, as a precautionary measure to prevent a spike in current flowing the wrong way, which would burn out the coil.

One obstacle I had to overcome was the fact that the voltage outputting from the 8051 is 3.3V while the relay requires 5V to operate. This was accomplished by adding a 2N3904 transistor to the switch, and connecting the ground connection from the relay to the collector, connecting the 8051 to

the base, and connecting the emitter to the switch's ground connection. This allows the relay to obtain the 5V from the switch's power supply, and obtain a ground connection when the 8051 is activated, therefore completing the circuit, and switching the relay.

With everything correctly implemented, the client can indeed select the appropriate scenario, which forces the server to launch the corresponding MPEG slide, activate the 8051 microcontroller, which changes each switch from Input A to Input B. When all is done, the system can correctly be reset. Therefore, this system functions exactly the way our client desired it to, with ease of use, low cost for implementation, and security and reliability. A sketch of the final circuit design can be seen below in Figure 04.

Figure 04: Sketch of Final Circuit Implementation



Performance Estimates and Results

The way this project was implemented allows for full functionality, with the client able to remotely launch the system and take over the television and radio stations with emergency broadcasting.

The most important variable in this system was making sure that the input levels, without passing through the hardware switch match that of the output when going through the switch. To make sure this was the case; measurements of the attenuation of the switch were obtained. As a reference, a standard composite video signal, which is used in this system, has a bandwidth of 4.2MHz, while audio has a bandwidth of 44.1 KHz. Using an oscilloscope (see Figure 05, below) and applying various frequencies to the IC in the switch using the function generator, I could obtain the approximate level when the output over the input becomes less than 1.0, causing peaking among the signals, leading to distortion. At approximately 9MHz, the output/input drops to 0.9 and at 12MHz, this begins curving downwards quite rapidly. Therefore, it can be concluded that the attenuation of the switch is normal up until at least 9MHz. Since a max of 4.2MHz is being sent for video and 44.1KHz for audio, which is less than half the bandwidth then the max supported by the IC, it can safely be said that there will be absolutely no distortion among signals passed through the switch. Figure 06, below, shows the attenuation of the IC.

Figure 05: Testing Environment

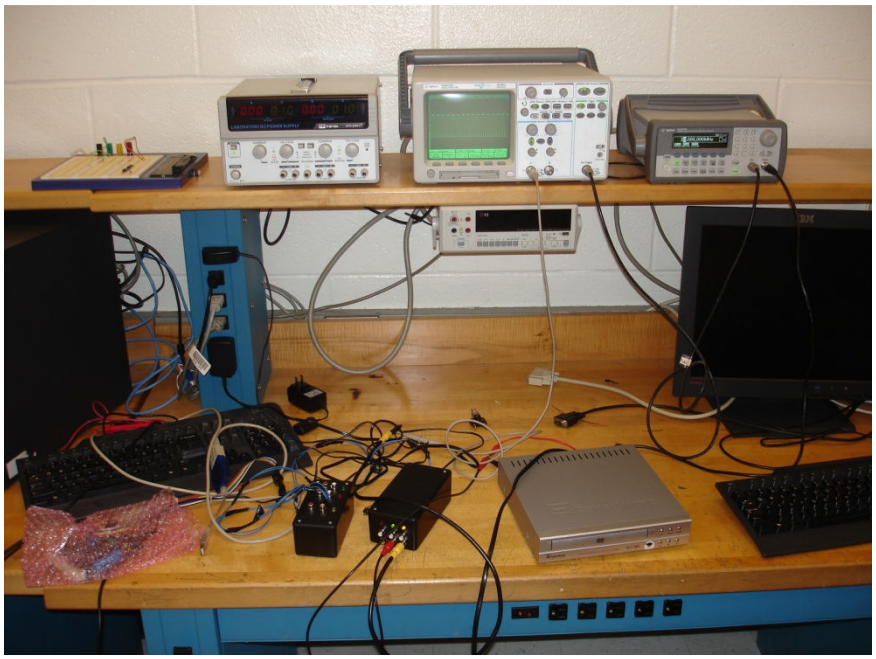
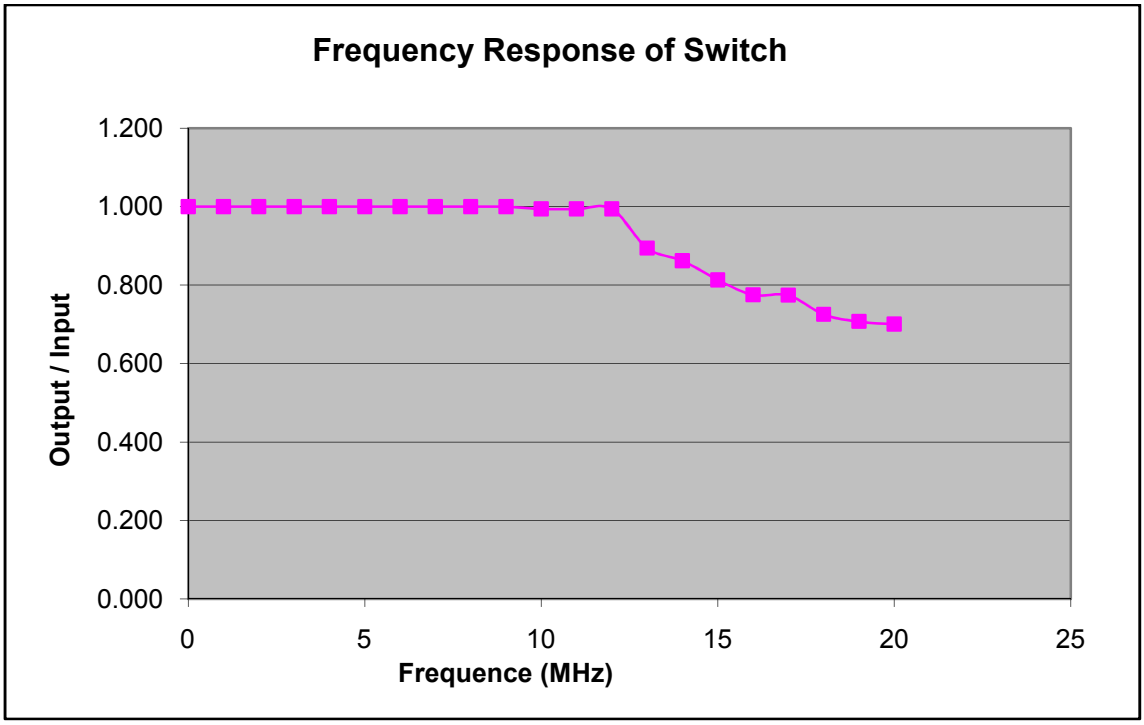


Figure 06: Graph of Frequency Response of Switch



In addition to the output levels, the entire system must activate 100% of the time and within a matter of seconds, as delay in an emergency situation can lead to more problems and added chaos. Upon testing just the hardware portion, the time it takes for the computer to send the activation command to the 8051 which switches the inputs is approximately 0.1 seconds, and appears nearly instantaneously to the eye. The computer takes approximately two seconds to launch the MPEG file once it receives the command from the client to launch the script. That being said, I am unable to give a reading for the time it takes to launch the software, as its development is not quite complete at this time. However, the time it takes from the time the client hits the "activate" button on their end to the time the emergency scenario is approximately 2-3 seconds, which is well within the range specified by the client.

Further research was spent analyzing the results of composite video passing through the oscilloscope. Figure 07, below, shows the waveform captured using the switch, while Figure 08 shows the waveform captured without using the switch. This, too, shows that the switch is indeed passing a correct composite video signal. Figure 08 shows a standard composite video waveform.

Figure 07: Waveform Captured w/o Switch

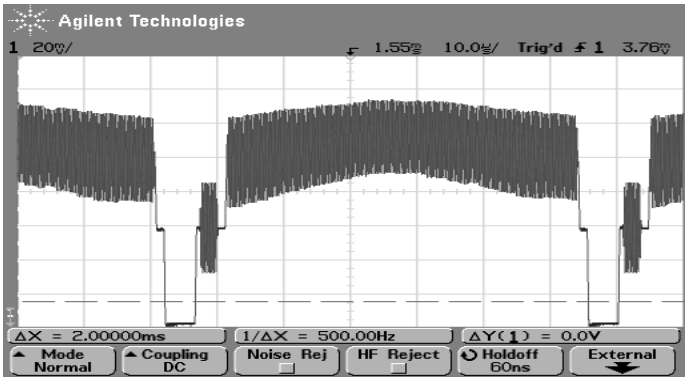


Figure 08: Waveform Captured w/ Switch

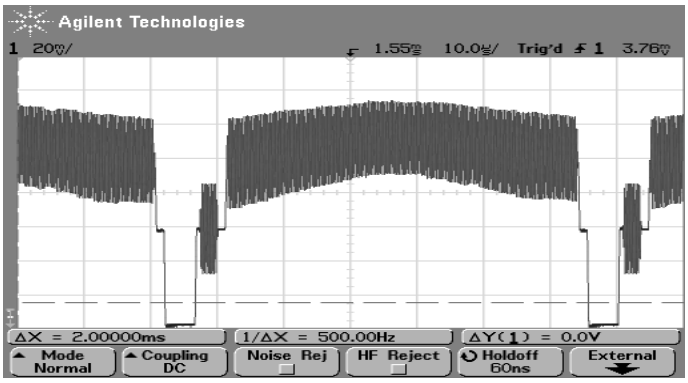
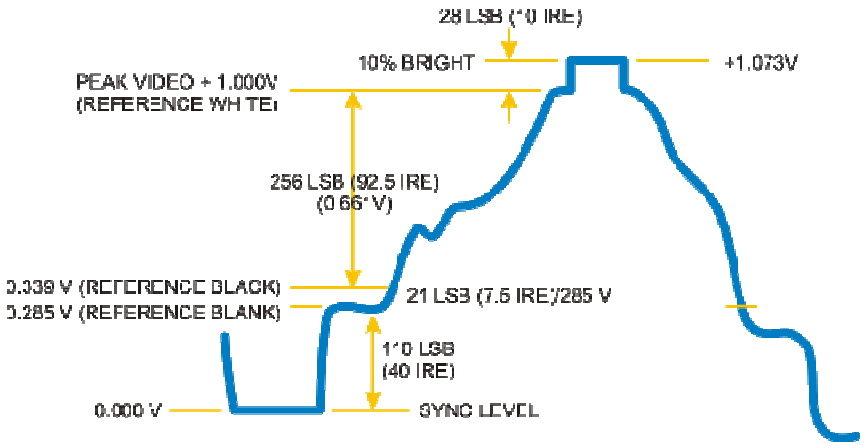


Figure 09: Composite Video Waveform



Production Schedule

Pre-Fall, 2007:

- Work with Campus Safety to establish design of system
- Research methods of implementation suitable to client's request and for senior project requirements
- Prepare TV and radio equipment for implementation of EAS

Fall, 2007:

- Week 1-2: Set up new TV studio - includes necessary internal wiring, configuration of computers, and equipment. This will prepare us for our new interconnections with TVuC and WRUC
- Week 3-4: Writing of IEF grant proposal to receive funding for equipment
- Week 5: Draft of presentation for project presentation
- Week 6: Dress-rehearsal of presentation and actual presentation given
- Week 7-8: Project research
- Week 9-10: Additional work on project paper and changes made to preliminary design

Winter Break:

- Web site
- Research for hardware to build switch with
- Research to allow 8051 to communicate with switch
- Build Linux PC which will act as the user interface, launch and display the video on the television and provide audio to both TV and radio

Winter, 2008:

- Week 1: Connected TVuC with WRUC; hardware implementation and changes at TVuC offices
- Week 2: Final research
- Week 3: Writing of IEF grant proposal to receive funding for equipment
- Week 4-7: Building of switches for TVuC; assisting in software development
- Week 8: Obtaining measurements, preparing presentation, presentation
- Week 9: Preparing MPEG design, additional measurements
- Week 10: User manual, final paper, final updates to Web site

Cost Analysis

The costs for this project are broken down into the following categories, with descriptions and approximate costs detailed:

- Emergency Computer: A computer must be dedicated to this project, as it will be needed to control the hardware switches and produce the emergency messages. The machine was donated by the TV studio with the only upgrade needed was a video card that had a TV-out connection. The machine runs Ubuntu, a build of the Linux operating system, chosen for its virtual 100% uptime. Other specifications include a Intel Celeron D processor, 1gb DDR2 memory, 120gb SATA hard drive, DVD-+RW drive. The system connects to the TVuC main rack which already included a KVM switch, allowing the computer to connect with the already existing monitor, keyboard and mouse.
    - $35 – Video Card
- Hardware Switches: The hardware switches for the EAS system were custom designed to work with our emergency computer. The switches connect various RCA jacks and interface them with an IC that controls what inputs get sent to the output jacks, based on the status of an 8051 microcontroller. This allows for the video to be forced from normal broadcasting to emergency broadcasting. Two of each component was needed, as two identical switches were built—one for TV and the other for radio.
    - $20 – Project Enclosures
    - $5 – RCA Jacks
    - $0 – Various Resistors, diodes, wiring, and transistors, provided by the Electrical Engineering department
    - $0 – 5V AC/DC converter box
    - $5 – IC
    - $0 – 8051 microcontroller, provided by the Computer Engineering department
    - $10 – Miscellaneous wiring, drill bits, and other minor components
- Backup Hardware Controller: At the beginning of this project, before any of the current implementation was set in stone, we obtained a Mini-T-NX audio/video controller unit. The function of this device is twofold. Firstly, it is used to replace the TV stations former controller which is used to interface all the video inputs with the single a/v modulator. The new one can be remotely controlled over a TCP/IP connection anywhere on the Union network, which helps the TV studio manage their broadcastings. In addition, it contains a hard drive and software which can store video and still slides on the device. Finally, it can create a detailed programming schedule and automatically switch inputs on its own. With regards to this project, Campus Safety is given a specific login to the software, in which they can launch an emergency scenario from there. This is designed as an backup alternative plan, should the original system go down or become unstable. Using this device requires proprietary software preinstalled on the user's system, as well as some training to launch the specific scenarios. Instructions on operating are located in the user's manual.
    - $2800 – Mini-T-NX (provided graciously by the Campus Safety department)

- Software: All software, including the logic to run the 8051 microcontroller, the client software and the server software, as well as the MPEG files used to display the emergency messages were custom designed by myself and my project partner.
    - $0 – Ubuntu OS
    - $0 – EAS System for 8051 Microcontroller
    - $0 – EAS Client software
    - $0 – EAS Server software

User's Manual

Since this project is not yet complete, due to the lack of software arising from my partner's absence, the User's Manual is not yet complete. However, Appendix B contains the User's Manual for the backup EAS system also implemented.

Discussion, Conclusions, and Recommendations

Now that this project is complete, I have provided my client, Union College's Campus Safety department with a method of alerting the campus community to an emergency scenario, should one arise on campus. With ease, a user can log into software on their own machine, locally select the scenario that applies to the emergency, and activate the system, providing views of both the television station and radio station with pertinent information. This, combined with other emergency notification systems already present improves safety overall on campus.

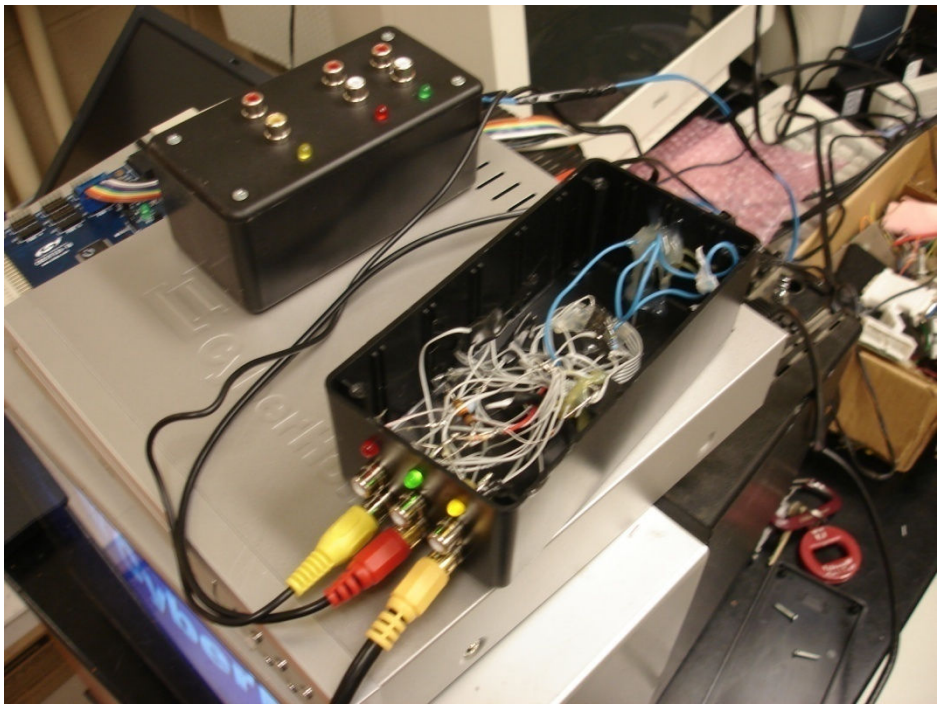
The goal of this project was to implement a system that would take over both the television and radio stations from a single remote source, with virtually complete uptime, low activation time, and ease of use so that users who are unfamiliar with the system can launch it. This project accomplishes all of these goals. The system should not crash or need to be rebooted, activates in less than two seconds, and is extremely user friendly. The trickiest part of the system, to the user, is remembering their login credentials. Once they are logged in, they can easily select one of the pre-recorded scenarios and hit launch, and the system will active. In addition, a backup plan has been implemented, albeit harder to operate. However, should one system go down, the other will still be available for usage.

For this system to be used to its full advantage does require that the user be somewhat familiar with its operation, so a brief skim-through of the User's Manual is recommended. In addition, it is recommended that Campus Safety have enough users trained to launch the system so that someone is always available to do it.

Future improvements, as always, can be made to this project. The first would be the implementation of better hardware switches (see Figure 10, below). Take the same physical hardware requirements, and package it into a printed circuit board. This decreases latency among the switching and improves reliability, as the internal connections will be more stable. In addition, a GUI for the

software will make user operation clearer. Finally, a long term improvement would be to modify this implementation such that a user could enter a custom message which would then be placed on the screen and read via text-to-speech. However, with any of these improvements, the system will still accomplish the same goal: alerting the campus to an emergency.

Figure 10: Internal Hardware Switch Wiring



Implementing this project has been a great educational endeavor for myself. Overall, I had to listen to a client who knew nothing about the technical implementation or limitations of a system and design a system that took all their requirements and satisfied them. Then, I had to figure out how to correctly wire up the hardware switches. This included soldering 16 wires to a surface-mount IC without getting any solder to touch adjacent pins. In addition, I had to figure out how to switch inputs from one source to another via an electronic signal. This involved familiarizing myself with relays and transistors, and implementing a microprocessor and necessary logic to accomplish this. Finally, and most

importantly, I learned how to verify that no signal degradation was occurring with my hardware. I needed to make sure that audio and video got passed through the switch without any loss whatsoever. Research prior to implementation showed that I should not have any loss, yet a faulty connection or an improper ground could lead to such conditions. However, tests using a signal generator and oscilloscope have proven that no loss occurs.

References and Links

Project Home Page, David A. Swift: http://antipasto.union.edu/~swiftd

Reference for Switch implementation, Engadget, March 13, 2007: http://www.engadget.com/2007/03/13/how-to-make-a-solid-state-a-v-switcher/

CBT3244A Datasheet, AllDataSheet.com: http://www.alldatasheet.com/datasheet-pdf/pdf/109603/PHILIPS/CBT3244A.html

Composite Video, Wikipedia: http://en.wikipedia.org/wiki/Composite_video

Video Circuits, Maxim-IC: http://pdfserv.maxim-ic.com/en/an/AN734.pdf

Student Projects, Electrical and Computer Engineering, Union College: http://ece.union.edu/projects/

Digi-Key Corporation (vendor): http://www.digikey.com

Newegg (vendor): http://www.newegg.com

Appendices

Appendix A: Code for 8051 Implementation

```
// ----------------------------------------------------------------

// Program: Emergency Alert System (for TVuC/WRUC @ Union College)
// Author: Dave Swift, Winter 08; assisted by Prof. James Hedrick

// NOTES: LED P1^6 causes UART 0 to stop working

// UART 0 uses P0 pin 0 for TX and pin 1 for RCV
// UART 1 uses P0 pin 2 for TX and pin 3 for RCV
// PORT P2 is used for DIO with pins 0 - 3 configured as outputs
//        and pins 4 - 7 configured as inputs

// ----------------------------------------------------------------

#pragma CODE
#include <c8051f020.h>
#include <stdio.h>
#include <string.h>

sfr16 DP               = 0x82;     // data pointer
sfr16 TMR3RL   = 0x92;     // Timer3 reload value
sfr16 TMR3     = 0x94;     // Timer3 counter
sfr16 ADC0     = 0xbe;     // ADC0 data
sfr16 ADC0GT   = 0xc4;     // ADC0 greater than window
sfr16 ADC0LT   = 0xc6;     // ADC0 less than window
sfr16 RCAP2    = 0xca;     // Timer2 capture/reload
sfr16 T2               = 0xcc;     // Timer2
sfr16 RCAP4    = 0xe4;     // Timer4 capture/reload
sfr16 T4               = 0xf4;     // Timer4
sfr16 DAC0     = 0xd2;     // DAC0 data

#define SYSCLK         22118400// SYSCLK frequency in Hz
#define BAUDRATEU0  9600    // Baud rate of UART0 in bps

void initialize(void);
void SYSCLK_Init(void);
void config(void);
void sync_pulse(void);
void UART0_Init(void);

sbit LED               = P1^6;     // LED built in development board
sbit TVUC_OUT  = P2^0;     // 0 = normal program; 1 = emergency
sbit WRUC_OUT  = P2^1;     // 0 = normal program; 1 = emergency
sbit TVUC_GND  = P2^2;        // used as a ground connection for TV
sbit WRUC_GND  = P2^3;        // used as a ground connection for RADIO
int chrs_read;
char in_buf[1];

void main(void)
{
       initialize();

       for(;;)
       {
              printf("!! CLEAR SIGNAL RECEIVED !!\n");
              TVUC_OUT = 0; // disable emergency system for TV
              WRUC_OUT = 0; // disable emergency system for radio
              LED = 0;

              chrs_read = scanf("%s",in_buf); // scan for data
              while (in_buf[0] == 'a')
              {
              printf("!! EMERGENCY SIGNAL RECEIVED !!\n");
                     TVUC_OUT = 1; // enable emergency system for TV
                     WRUC_OUT = 1; // enable emergency system for radio
```

```
                        LED = 1;

                        chrs_read = scanf("%s",in_buf); // scan for data
                }
        }
}

// -------------------------
// configuration subroutines
// -------------------------

// begin initialization subroutines
void initialize(void)
{
        TVUC_GND = 0;          // enable ground for TV
        WRUC_GND = 0;          // enable ground for radio
        WDTCN = 0xde;          // disable watchdog timer
        WDTCN = 0xad;
        config();                       // load configurations
        UART0_Init();          // initialize UART0
        SYSCLK_Init();         // initialize oscillaor

        printf("\n\n------ SYSTEM RESET and Active ------");
        printf("\n\nWelcome to the Union College Emergency Alert System for TVuC and WRUC\n\n");
}

// configured using Silicon Industries Configuration Wizard
void config(void)
{
        //configure the XBRn Registers
        XBR0 = 0x04; //XBAR0: Initial Reset Value
        XBR1 = 0x00; //XBAR1: Initial Reset Value
        XBR2 = 0x44; //XBAR2: Initial Reset Value

        //port configuration (1 = Push Pull Output)
        P0MDOUT = 0x05; //output configuration for P0
        P1MDOUT = 0x40; //output configuration for P1
        P2MDOUT = 0x0F; //output configuration for P2
        P3MDOUT = 0x00; //output configuration for P3
        P74OUT = 0x00;  //output configuration for P4-7
        P1MDIN = 0xFF;  //Input configuration for P1
}

// this routine initializes the system clock to use an 22.1184 MHz crystal as its clock source
void SYSCLK_Init(void)
{
        int i;                                          //delay counter
        OSCXCN = 0x67;              //start external oscillator with //22.1184 MHz crystal
        for (i=0; i<256; i++);      //wait for oscillator to start
        while(!(OSCXCN & 0x88));    //wait for crystal oscillator to settle
        OSCICN = 0x88;              //select external oscillator as SYSCLK
                                                                //source and enable mission clock
detector
}

// configure the UART0 using Timer1, for <baudrate> and 8-N-1
// UART 0 uses P0 pin 0 for TX and P0 pin 1 for RCV
void UART0_Init(void)
{
        SCON0 = 0x50;                           //SCON0: mode 1, 8-bit UART, enable RX
        TMOD = 0x20;                            //TMOD: Timer1, mode 2, 8-bit reload
        TH1 = -(SYSCLK/BAUDRATEU0/16);//set Time1 reload value for baudrate
        TR1 = 1;                                        //start Timer1
        CKCON |= 0x10;                          //Timer1 uses SYSCLK as time base
        PCON |= 0x80;                           //SMOD00 = 1 (disable baudrate divde-by-two
        TI0 = 1;                                        //Indicate TX0 ready
}
```

Appendix B: Backup EAS System User's Manual

This is an outline. Look for the final User's Manual coming soon…

Launching System

Backup Plan 01

- Manually launch from TV station

Backup Plan 02

- Manually launch using TVuC controller
    - User name: Safety
    - Password: Campus
    - IP address: tvuc.union.edu
    - Procedure:
        - Launch "Mini-T-NX" software from desktop
        - Log in
        - Click on window at top
        - Etc
    - When complete, return broadcasting to normal scenario

Should a problem arise with the hardware switches

Should the software not connect, make sure that the connection from the client to the server is available by running a ping command towards "tvuc.union.edu". To do this, click Start >> Run and type in "cmd" and hit OK. Then at the "C:\.....>" prompt, enter "ping tvuc.union.edu". If the system returns "request timed out", contact ITS to determine why the connection has gone down. If it goes through, try restarting your machine. If this too fails, the system in Reamer in the TV offices must be reset. Please contact the president or technical manager of the TV station to do this.